International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 3, No. 4
November 2014

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 3, No. 4, November 2014

Research Paper

IMPLEMENTATION OF DIGITAL CMOS COMPARATOR USING PARALLEL PREFIX TREE

Nagaswapna Manukonda[1]* and H Raghunadh Rao[2]

*Corresponding Author: ***Nagaswapna Manukonda*** ✉ *swapnabtec@gmail.com*

Comparators are the key design elements for a wide range of applications like scientific computation (graphics and image/signal processing), test circuit applications (jitter measurements, signature analyzers, and built-in self test circuits) and for general-purpose processor components (associative memories, load-store queue buffers, translation look-aside buffers, branch target buffers) and many other CPU argument comparison blocks. In this project a 16,32,64 bit comparator architectures is designed by using parallel prefix structure. This project evaluates the successful results as per requirement and specifications. In existing system, the parallel prefix structure is designed for 16, 32 and 64 bit architectures and the reports from the Xilinx tool concludes that for every bit range doubles the delay, memory, LUT and power has not doubled up to the mark. But In the proposed design of my project, each and every element in the parallel prefix structure will be replaced by universal logic (multiplexer) and the obtained results will be compared with existed design for the same device specifications. By performing this modification in the architecture will leads to reduction in POWER CONSUMPTION and in DELAY parameters.

Keywords: Parallel prefix tree structure, Bitwise competition logic (BCL)

INTRODUCTION

Other comparator designs improve scalability and reduce comparison delays using a hierarchical prefix tree structure composed of 2-b comparators. These structures require log2 N comparison levels, with each level consisting of several cascaded logic gates. However, the delay and area of these designs may be prohibitive for comparing wide operands. The prefix tree structure's area and power consumption can be improved by leveraging two-input multiplexers (instead of 2-b comparator cells) at each level and generate-propagate logic cells on the first level (instead of 2-b adder cells), which takes advantage of one's complement addition. Using this logic composition, a prefix tree requires six levels for the most common comparison bit width of 64 bits, but suffers from high power

[1] M.Tech. Student, Department of ECE, Chirala Engineering College, Chirala 523155, Prakasam Dt., AP.

[2] Associate Professor, Department of ECE, Chirala Engineering College, Chirala 523155, Prakasam Dt., AP.

consumption due to every cell in the structure being active, regardless of the input operands' values.

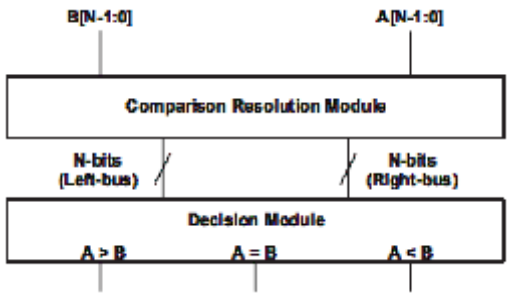
Furthermore, the structure can perform only "greater-than" or "less-than" comparisons and not equality. To improve the speed and reduce power consumption, several designs rely on pipelining and power-down mechanisms to reduce switching activity with respect to the actual input operands' bit values. One design uses all-N transistor (ANT) circuits to compensate for high fan-in with high pipeline throughput. A 64-b comparator requires only three pipeline cycles using a multiphase clocking scheme. However, such a clocking scheme may be unsuitable for high-speed single-cycle processors because of several heavily loaded global clock signals that have high-power transition activity. Additionally, race conditions and a heavily constrained clock jitter margin may make this design unsuitable for wide-range comparators.

An alternative architecture leverages priority-encoder magnitude decision logic with two pipelined operations that are triggered at both the falling and rising clock edges to improve operating speed and eliminate long dynamic logic chains. However, 64-b and wider comparators require a multilevel cascade structure, with each logic level consisting of seven nMOS transistors connected in series that behave in saturating mode during operation. This structure leads to a large overall conductive resistance, with heavily loaded parasitic components on the clock signal, which severely limits the clock speed and jitter margin. Other architectures use a multiplexer-based structure to split a 64-b comparator into two comparator stages: the first stage consists of eight modules performing 8-b comparisons and the modules' outputs are input into a priority encoder and the second stage uses an 8-to-1 multiplexer to select the appropriate result from the eight modules in the first stage.

Similarly, other energy-efficient designs leverage schemes to reduce switching activity. Compute-on demand comparators compare two binary numbers one bit at a time, rippling from the most significant bit (MSB) to the least significant bit (LSB). The outcome of each bit comparison either enables the comparison of the next bit if the bits are equal, or represents the final comparison decision if the bits are different. Thus, a comparison cell is activated only if all bits of greater significance are equal. Although these designs reduce switching, they suffer from long worst case comparison delays for wide worst case operands. To reduce the long delays suffered by bitwise ripple designs, an enhanced architecture incorporates an algorithm that uses no arithmetic operations. This scheme detects the larger operand by determining which operand possesses the leftmost 1 bit after pre-encoding, before supplying the operands to bitwise competition logic (BCL) structure. The BCL structure partitions the operands into 8-b blocks and the result for each block is input into a multiplexer to determine the final comparison decision. Due to this BCL-based design's low transistor count, this design has the potential for low power consumption, but the pre-encoder logic modules preceding the BCL modules limit the maximum achievable operating frequency. In addition, special control logic is needed to enable the BCL units to switch dynamically in a synchronized fashion, thus increasing the power consumption and reducing the operating frequency.

To alleviate some of the drawbacks of previous designs (such as high power consumption, multi cycle computation, custom structures unsuitable for continued technology scaling, long time to market due to irregular VLSI structures, and irregular transistor geometry sizes), in this paper we leverage standard CMOS cells to architect fast, scalable, wide-range, and power-efficient algorithmic comparators with the following key features.

Figure 1: Block Diagram of our Comparator Architecture, Consisting of a Comparison Resolution Module Connected to a Decision Module

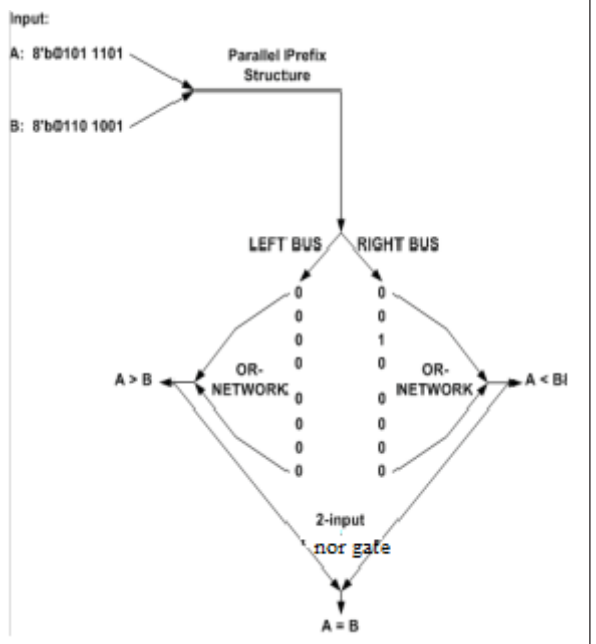


COMPARATOR ARCHITECTURAL OVERVIEW

The comparison resolution module in Fig. 1 (which depicts the high-level architecture of our proposed design) is a novel MSB-to-LSB parallel-prefix tree structure that performs bitwise comparison of two N-bit operands A and B, denoted as AN"1, AN"2, . . ., A0 and BN"1, BN"2, . . ., B0, where the subscripts range from N–1 for the MSB to 0 for the LSB. The comparison resolution module performs the bitwise comparison asynchronously from left to right, such that the comparison logic's computation is triggered only if all bits of greater significance are equal.

The parallel structure encodes the bitwise comparison results into two N-bit buses, the left bus and the right bus, each of which store the partial comparison result as each bit position is evaluated, such that

Figure 2: Example 8-b Comparison



if Ak > Bk, then leftk = 1 and rightk = 0

if Ak < Bk , then leftk = 0 and rightk = 1

if Ak = Bk , then leftk = 0 and rightk = 0.

In addition, to reduce switching activities, as soon as a bitwise comparison is not equal, the bitwise comparison of every bit of lower significance is terminated and all such positions are set to zero on both buses, thus, there is never more than one high bit on either bus. The decision module uses two OR-networks to output the final comparison decision based on separate OR-scans of all of the bits on the left bus (producing the L bit) and all of the bits on the right bus (producing the R bit). If LR = 00, then A = B, if LR = 10 then A > B, if LR = 01 then A < B, and LR = 11 is not possible. An 8-b comparison of input operands A = 01011101 and B = 01101001 is

illustrated in Figure 2. In the first step, a parallel prefix tree structure generates the encoded data on the left bus and right bus for each pair of corresponding bits from A and B.

In the above example, A7 = 0 and B7 = 0 encodes as left7 = right7 = 0, A6 = 1, and B6 = 1 encodes as left6 = right6 = 0, and A5 = 0 and B5 = 1 encodes left5 = 0 and right5 = 1. At this point, since the bits are unequal, the comparison terminates and a final comparison decision can be made based on the first three bits evaluated. The parallel prefix structure forces all bits of lesser significance on each bus to 0, regardless of the remaining bit values in the operands. In the second step, the OR-networks perform the bus OR-scans, resulting in 0 and 1, respectively, and the final comparison decision is A > B. We partition the structure into five hierarchical prefixing sets, as depicted in Figure 3, with the associated symbol representations in Tables I and II, where each set performs a specific function whose output serves as input to the next set, until the fifth set produces the output on the left bus and the right bus.

The below symbols are usually used in implementation. Each symbol is represented by the corresponding logic gates. The symbol will perform the operation represented by the logic gate and maximum fan in and fan outs are indicated as 2/4 I.e., the maximum number of inputs are 2 and the maximum number of outputs are 4. These symbols are used to implement the several sets of operations.

Table 1: Symbol Notation and Definitions

Symbol (Cells)	Definition
N	Operand bitwidth
A	First input operand
B	Second input operand
R	Right bus result bit
L	Left bus result bit
$\prod$	Bitwise AND
$\sum$	Bitwise OR
T{*}	Logic function of cell type*
COMP{*}	Complement function of set*[i]

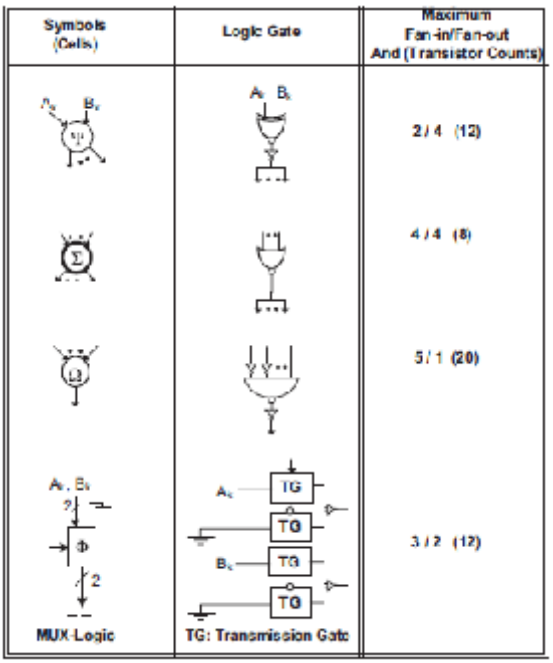
All cells (components) within each set operate in parallel, which is a key feature to increase operating speed while minimizing the transitions to a minimal set of leftmost bits needed for a correct decision. This prefixing set structure bounds the components' fan-in and fan-out regardless of comparator bit width and eliminates heavily loaded global signals with parasitic components, thus improving the operating speed and reducing power consumption. Additionally, the OR-network's fan-in and fan-out is limited by partitioning the buses into 4-b groupings of the input operands, thus reducing the capacitive load of each bus.

COMPARATOR DESIGN DETAILS

We partition the structure into five hierarchical prefixing sets, as depicted in Fig.2 with the associated symbol representations in Tables I , where as each set performs a exact function whose output serves as input to the next set, in hope of the fifth set produces the output on the left bus and the right bus Every part of cells components within each set operate in parallel were as it's a key feature to increase operating speed while minimizing the transitions to a minimal set of left most bits needed for a correct decision. This prefixing set structure bounds the components fan-in and fan-out regardless of comparator bit-width and eliminates heavily loaded global signals with parasitic components, thus improving the operating speed and reducing power consumption.

In this section, we detail our comparator's design Figure 2, which is based on using a novel parallel prefix tree Tables 1 and 2 contain symbols and definitions. Each set or groups of cells that produce output and serve as inputs to the next set in the hierarchy, with the exception of set 1, the outputs serve as inputs to several sets. Set 1 compares the N -bit operands A and B bit-by-bit, using a single level of N * Type cell. The * type cells provide a

Figure 3: Logic Gate Representations for Symbols



Termination flag Dk to cells in sets 2 and 4, indicating whether the computation should terminate. These cells compute (where $0 \le k \le N - 1$) .

$$Dk = Ak \oplus Bk$$

BASIC ARCHITECTURE 0F 16 BIT COMPARATOR USING PARALLEL PREFIX TREE

For an Ω type cell and the 4-b partition to which the cell belongs, bitwise comparison outcomes from set 1 provide information about the more significant bits in the cell's Ω type cells, Set 5 consists of N Φ -type cells (two-input, 2-b-wide multiplexers). One input is (AK, Bk) and the other is hardwired to "00." The select control input is based on the Ω type cell output from set 4. We define the 2-b as the left-bit code (AK) and the right-bit code (Bk), where all left-bit codes and all right-bit codes combine to form the left bus and the right bus, respectively. The Φ-type cells compute (where $0 \le k \le N - 1$).

$$\Phi = f_k^{1,0} = y_k \times m_k + \bar{y}k \times (00)$$

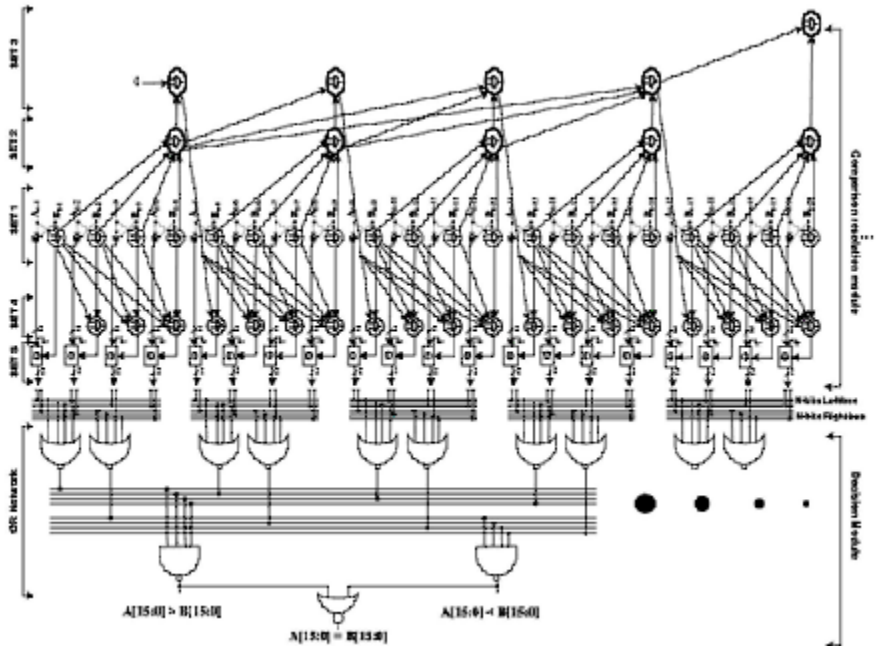
The output $f_k^{1,0}$ denotes the "greater-than," "less-than," or "equal to" final comparison decision.

$$F_k^{1,0} \begin{cases} 00, & for\ A_k = B_k \\ 01, & for\ A_k = B_k \\ 10, & for\ A_k = B_k \end{cases}$$

Essentially, the 2-b code $f_k^{1,0}$ can be realized by OR-ing all left bits and all right bits separately, as shown in the decision module, using an OR-gate network in the form of NOR-NAND gates yielding a more optimum gate structure

We define the 2-b as the left-bit code (Ak) and the right-bit code (Bk), where all left-bit codes and all right-bit codes combine to form the left bus and the right bus, respectively. The Φ-type cells compute (where $0 \le k \le N - 1$. From left to right, the first four Σ^3-type cells in set 3 combine the 4-b partition comparison outcomes from the one, two, three, and four 4-b partitions of set 2. Since the fourth Σ^3-type cell has a fan-in of four, the number of levels in set 3 increases and set 3's fifth Σ^3-type cell combines the comparison

Figure 4: Implementation Details for the Comparison Resolution Module (Sets 1 Through 5) and the Decision Module



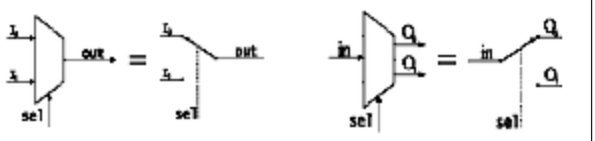
outcomes of the first 16 MSBs with a fan-in of only two and a fan-out of one.

PROPOSED ARCHITECTURE

Replacing With Multiplexer Logic

In this project the switching logic and the main block design is carried out by using mux logic to perform low power operations because , In electronics, a multiplexer (or mux) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output. Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. A multiplexer is also called a data selector. An electronic multiplexer makes it possible for several signals to share one device or resource, for example one A/D converter or one communication line, instead of having one device per input signal. An electronic multiplexer can be considered as a multiple-input, single-output switch.

Figure 5: Switch



ADVANTAGES BY USING MULTIPLEXR BASED IN PARALLEL PREFIX TREE

Low power consumption by replacing the needed logics by multiplexer, because multiplexer operates at very low power switching transitions compared to buffer and other logical gates. Low delay compared to normal based comparator,

which in turn defines the high speed operations. Less area consumption in terms of number of slices. Less number of Lut's compared to existing approach.

Figure 6: Internal Schematic for Comparator for 64 Bit

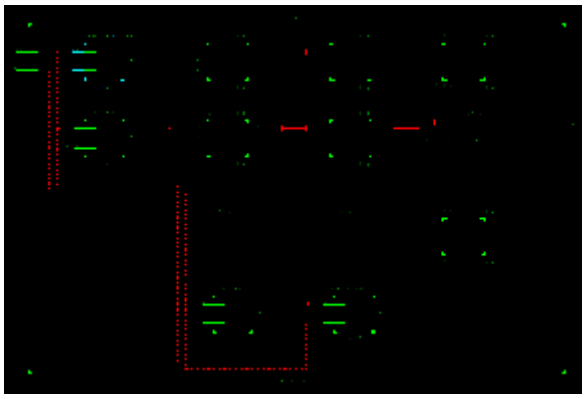


Figure 7: Simulation Result

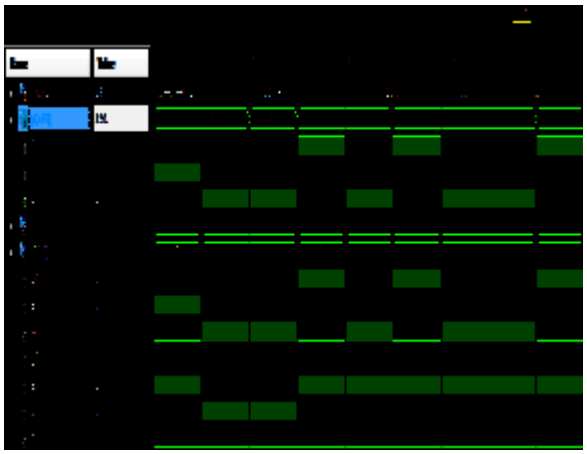


Table 2: comparision of 16 32 and 64 bit Delay and Power values of scalable comparator

BITS DELAY (nsec)	POWER(mw)			
	Existing	Proposed	Existing	Proposed
16	16.5	12.6	0.46	0.35
32	16.2	13.9	0.96	0.78
64	20.0	17.4	1.94	1.57

CONCLUSION

In this project a scalable high-speed low-power Comparator designed using regular digital hardware structures consisting of two modules: the comparison resolution module and the decision module. These modules are structured as parallel prefix trees with repeated cells in the form of simple stages that are one gate level deep with a maximum fan out of 2.17, independent of the input bit width. This regularity allows simple prediction of comparator characteristics for arbitrary bit widths and is attractive for continued technology Scaling and logic synthesis.

These modules are structured as parallel prefix trees by using a normal flow. But in normal tree structure a delay of 16.5nsec, 16.2nsec , 20.0nsec and power of 0.47mw, 0.96mw,1.94 mw has observed for 16,32,64 bit sized comparators, but here a proof has made that by using multiplexer based technique the delay of 12.6nsec, 13.9nsec , 17.4nsec and power of 0.35mw, 0.78mw,1.57 mw has reduced by using simulation on Xilinx under the device of xc3s500e-4fg320.In future scope ,the comparator bit size can increased to 128 considering the fear factors like delay and power as a important factors ,which should not be increased so that it can affect the performance of the comparator . Future work will include additional circuit optimizations to further reduce the power dissipation by adapting dynamic and analog implementations for the comparator resolution module and a high-speed zero-detector circuit for the decision module. Given that our comparator is composed of two balanced timing modules, the structure can be divided into two or more pipeline stages with balanced delays, based on a set structure, to effectively increase the comparison throughput.

REFERENCES

1. Abramovici M, Breuer M A, and Friedman A D (1990), *Digital Systems Testing and Testable Design,* Piscataway, NJ: IEEE Press.
2. Chan A H and Roberts G W (2004), "A jitter characterization system using a component-invariant Vernier delay line," IEEE Trans. Very Large Scale Integr. (VLSI) *Syst.*, Vol. 12, No. 1, pp. 79–95.
3. Helms H L (1989), *High Speed (HC/HCT) CMOS Guide,* Englewood Cliffs, NJ: Prentice-Hall.
4. Liu H J R and Yao H (1998), *High*-Performance VLSI Signal Processing *Innovative Architectures and Algorithms*, Vol. 2. Piscataway, NJ: IEEE Press.
5. Oklobdzija V G (1994), "An algorithmic and novel design of a leading zero detector circuit: Comparison with logic synthesis," IEEE Trans. Very Large Scale Integr. (VLSI) *Syst.*, Vol. 2, No. 1, pp. 124–128.
6. Parhami B (2009), "Efficient hamming weight comparators for binary vectors based on accumulative and up/down parallel counters," *IEEE Trans. Circuits Syst.*, vol. 56, no. 2, pp. 167-171.
7. Ponomarev D V, Kucuk G, Ergin O and Ghose K (2004), "Energy efficient comparators for superscalar data paths," *IEEE Trans. Comput.,* Vol. 53, No. 7, pp. 892–904.
8. Sheng Y and Wang W (2008), "Design and implementation of compression algorithm comparator for digital image processing on component," *in Proc. 9th Int. Conf. Young Comput. Sci.*, November, pp. 1337–1341.
9. SN7485 4-bit Magnitude Comparators, Texas Instruments, Dallas, TX, 1999.
10. Suzuki H, Kim C H, and Roy K (2007), "Fast tag comparator using diode partitioned domino for 64-bit microprocessor," *IEEE Trans. Circuits Syst. I,* Vol. 54, No. 2, pp. 322–328.

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com